A VISUAL BEGINING TO UNDERSTANDING CONCURRENCY

Concurrency

Concepts

Mark Moeykens

AN VISUAL REFERENCE GUIDE FOR DEVELOPERS

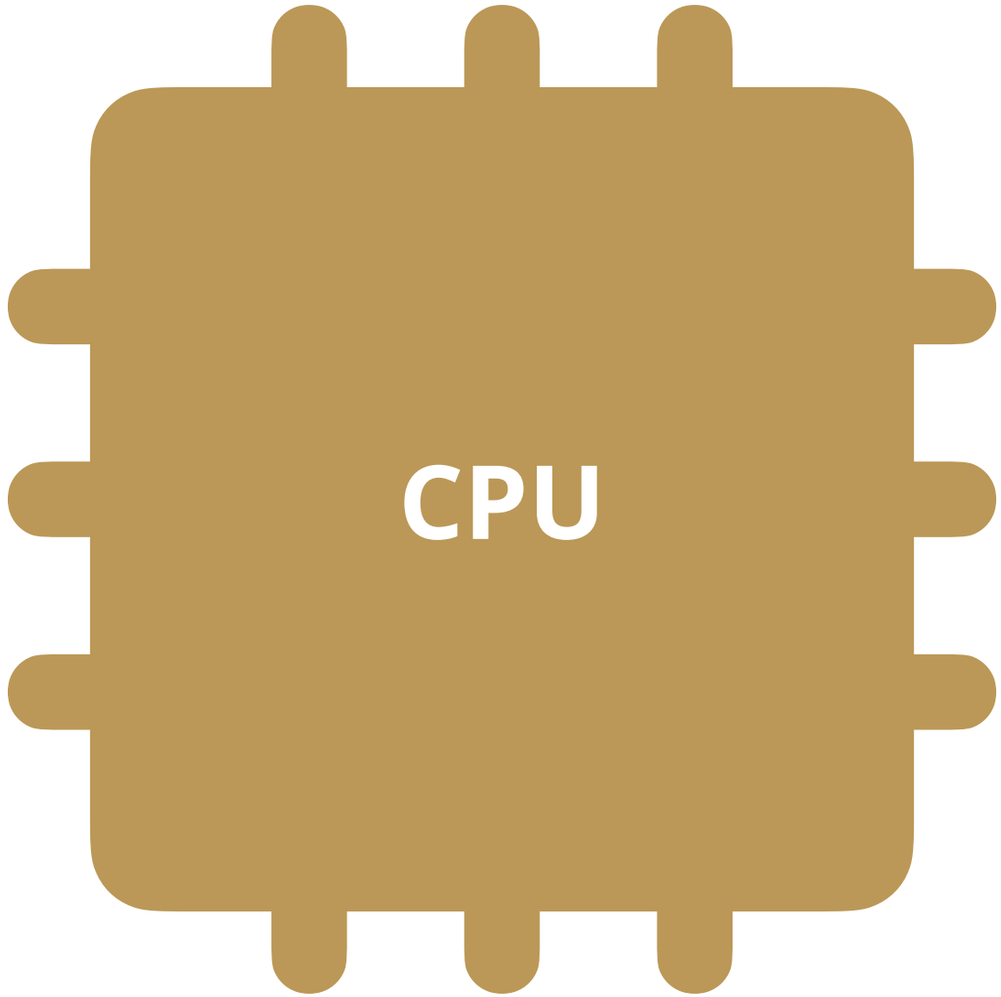
Big Mountain Studio

Version: 20-NOVEMBER-2021

CONCEPTS & TERMS

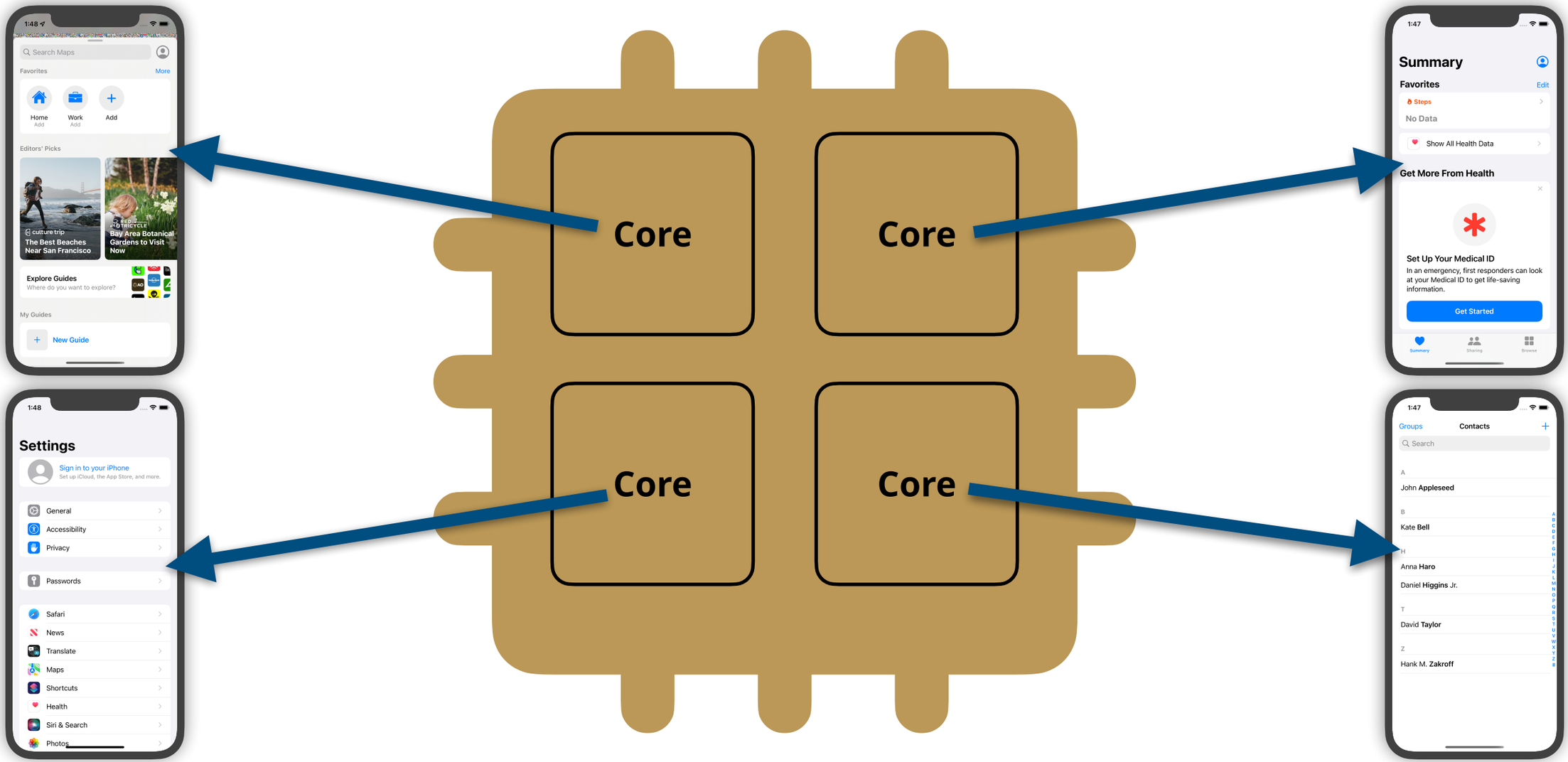
Doing Work

A central processing unit (CPU) is a physical device that handles all instructions it receives from hardware and software. This is a physical chip within your devices. It receives tasks from your apps and executes them.



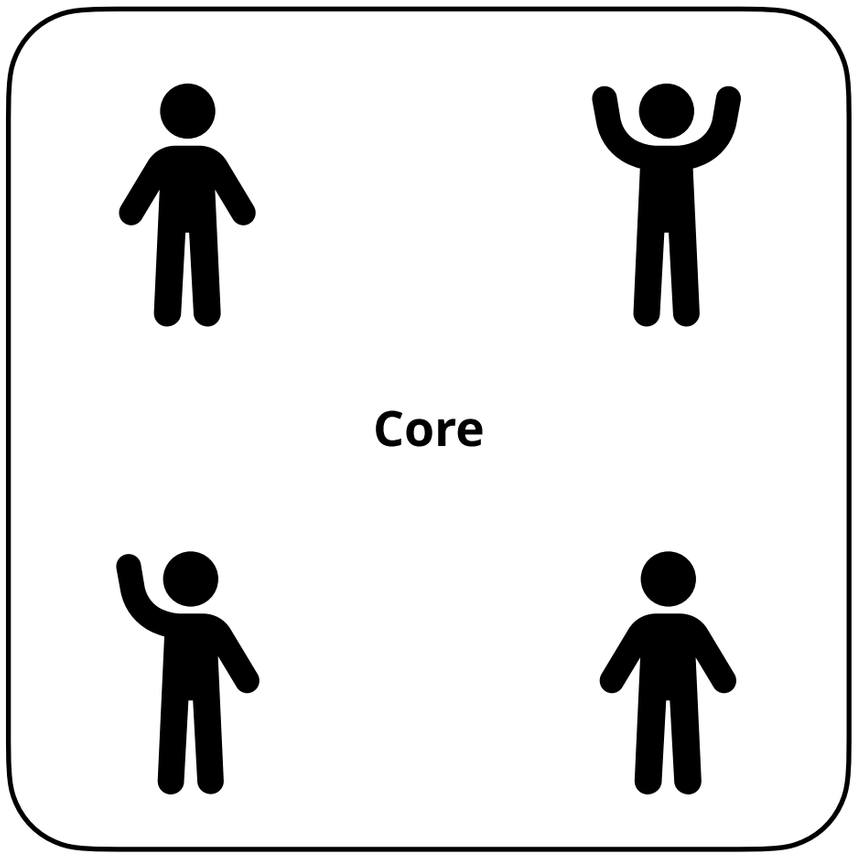
Handling Work with Cores

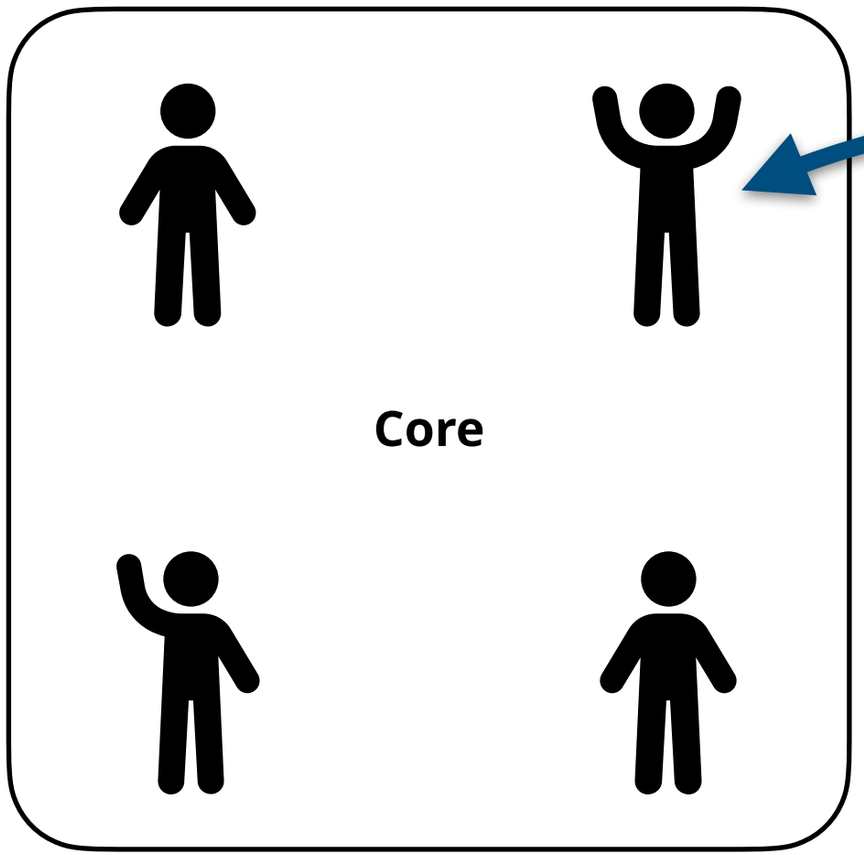
A CPU can be divided up into many cores. Cores are the part of the CPU that actually executes your app's tasks. Multiple cores allow the device to run multiple apps at the same time too.



Handling Work with Threads

Threads execute tasks assigned to them. Think of threads as people on your CPU. You can assign work to threads/people. The threads could be on the same core or different cores.





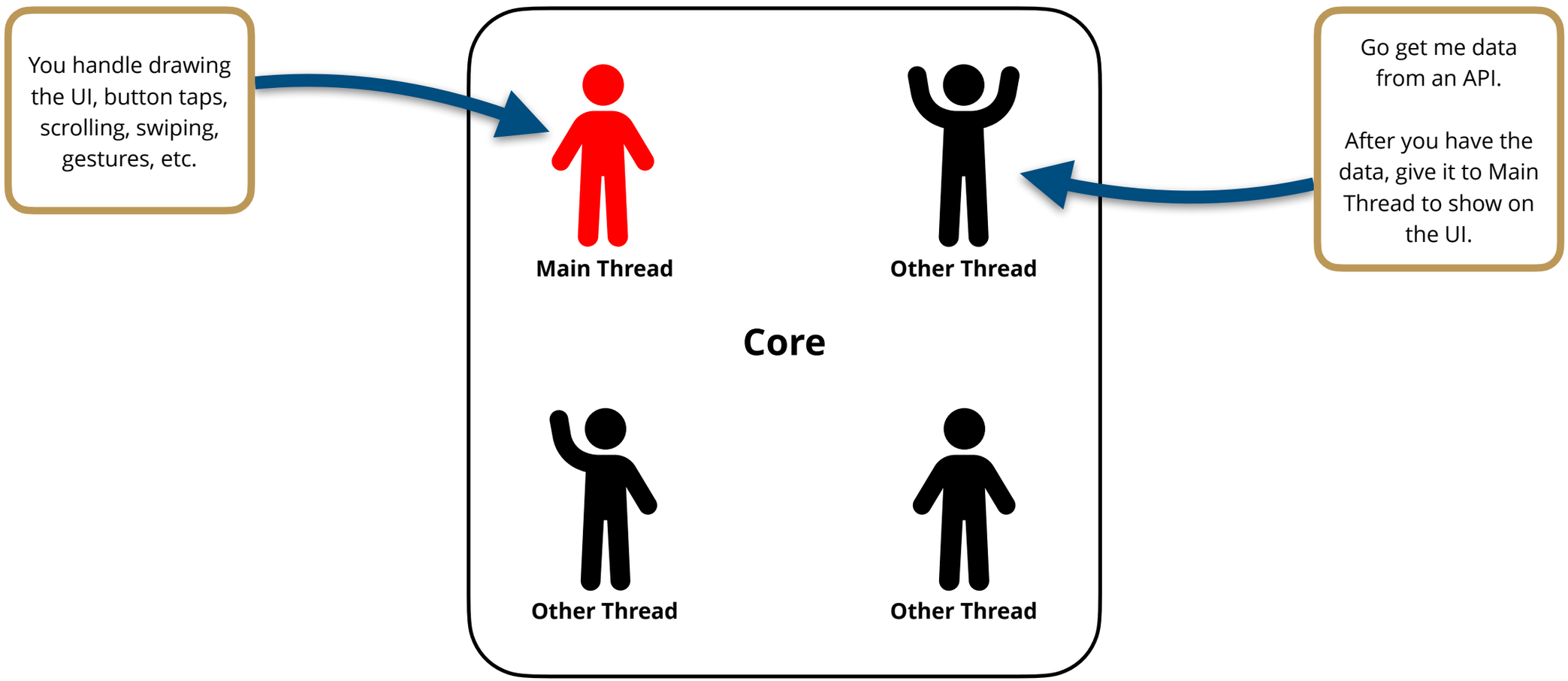
Each person/thread performs tasks. **A task is a unit of work.**

Think of tasks as a number of items you want to perform in your app:

1. Disable button after tapping
2. Get data
3. Show animation
4. Show progress indicator
5. Show data

Main Thread

Every app has a “main thread”. This is the thread where the UI work is done, such as drawing the UI, handling scrolling or button taps. Other threads could be used to perform tasks in the background, such as getting data or performing intensive calculations.



HANDLING WORK

This next chapter will go over the terms and concepts for the three ways in which your code can run. **Yes, there are only three ways.**

Synchronous Work

1

Synchronous work means one task has to complete before the next task is started.

1

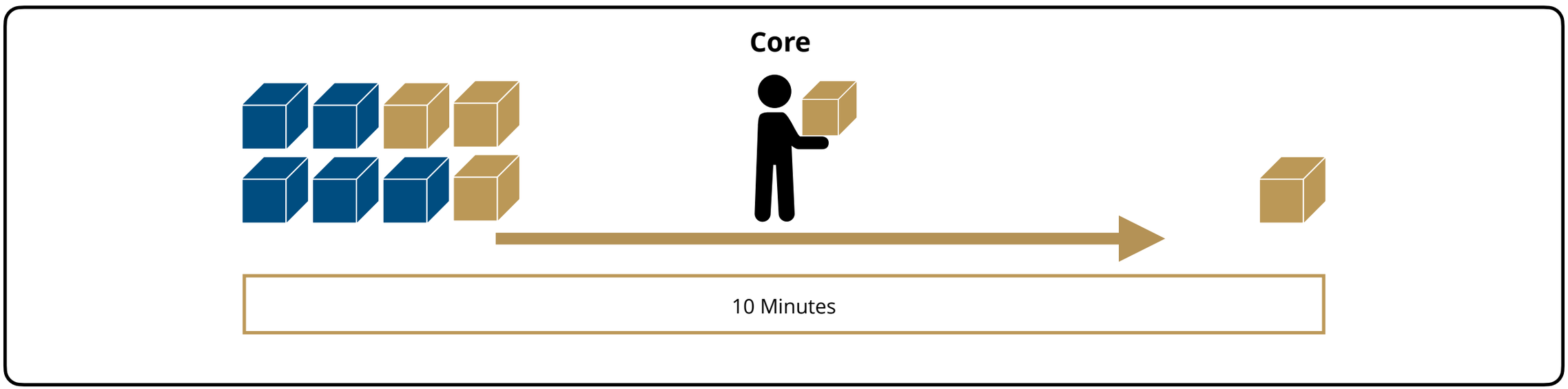
Synchronous Work

1

A person (thread) can perform tasks in different ways. One person can do one task at one time. They will not start the next task until the first task is done. This is called synchronous work because the second task will not be started until the first task is complete. This is normally how most of your code is run.

Tasks

1. Move gold boxes
2. Move blue boxes



Performance

Let's say it takes:

1. 5 minutes to move 5 gold boxes
2. 5 minutes to move 5 blue boxes

10 minutes total

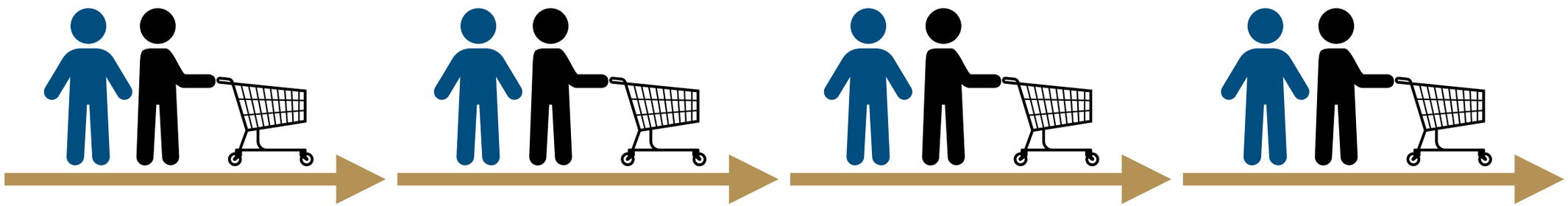
1

Synchronous Work Example 1

1

Another example of synchronous work: Say you and a friend or spouse go to the grocery store. You stay together while you perform the following tasks:

Store



Task: Get rice	Task: Get vegetables	Task: Get chicken	Task: Get cream
2 Minutes	2 Minutes	2 Minutes	2 Minutes

All shopping will be complete in **8 minutes**.
When looking at tasks this way, you might already be thinking of some ways you can make this work more efficient by dividing up the work.

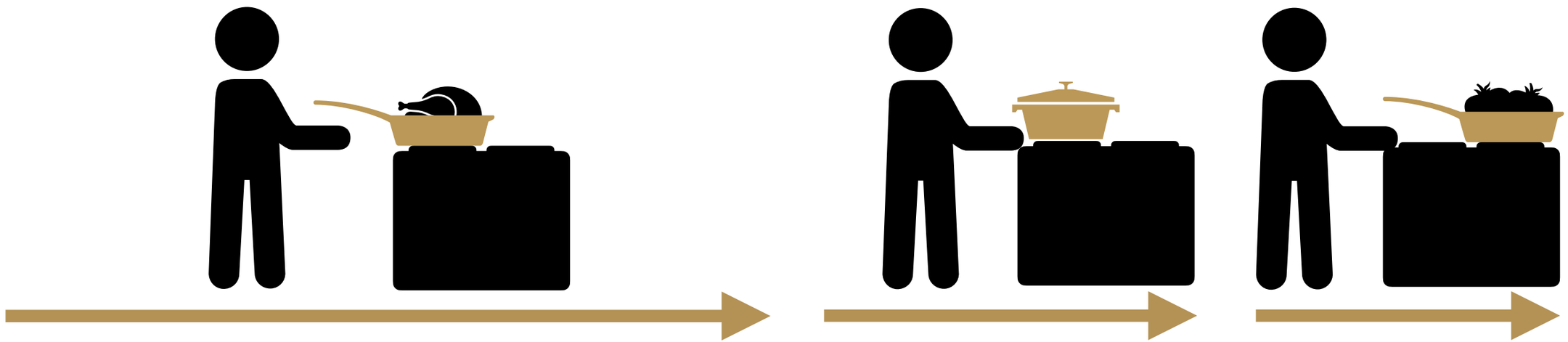
1

Synchronous Work Example 2

1

If you were to cook a meal synchronously, it might go like this:

Kitchen



Task: Cook chicken

30 Minutes

Task: Cook rice

15 Minutes

Task: Cook vegetables

10 Minutes

Cooking just one thing at a time will take you **55 minutes**.
You might already be thinking of some ways you can make this work more efficient by doing two or more things at one time.

Asynchronous Work

2

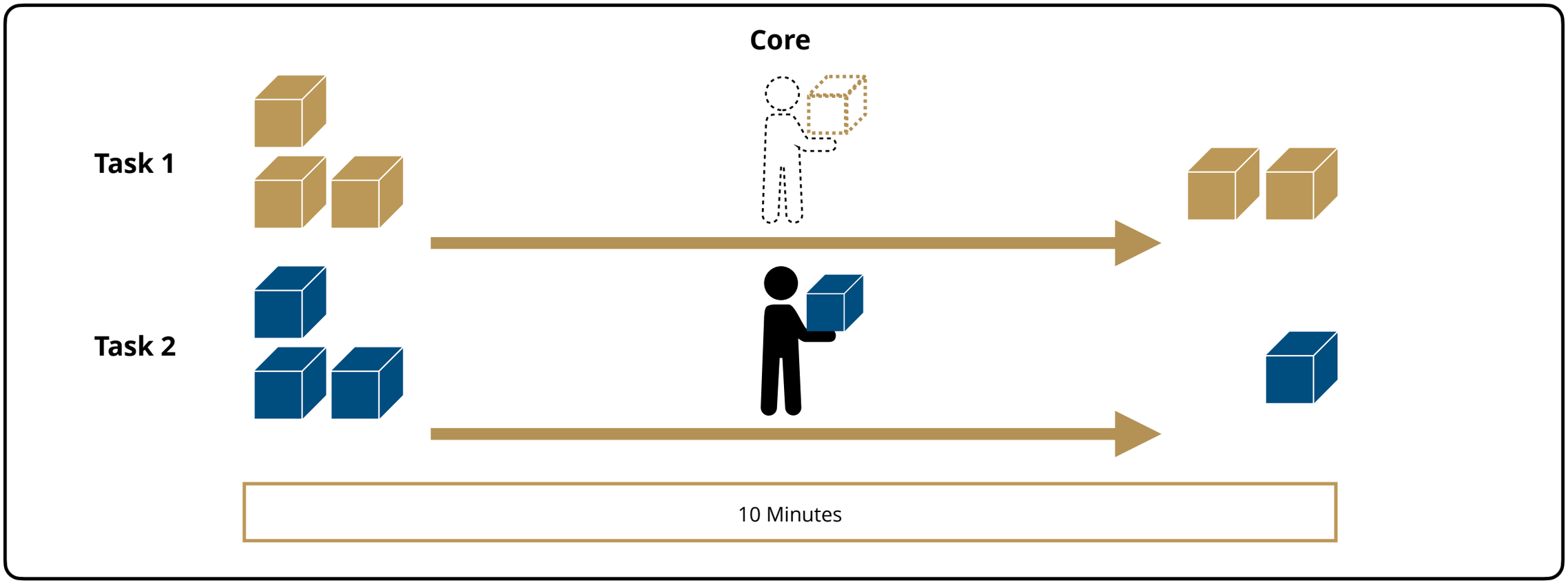
Asynchronous work means that the second task can start, even though the first task has NOT completed.
But as you will see, the two tasks do NOT happen at the same time.

2

Asynchronous Work

2

Asynchronous work is when one person splits up their time between many tasks. It is still only one person/thread. But the person can now do part of task 1 (one box) then do part of task 2 (one box), then go back to task 1, etc.



Note: The person/thread will suspend task 1 to work on task 2. Then suspend task 2 to work on task 1.

Performance

1. 5 minutes to move 5 gold boxes
2. 5 minutes to move 5 blue boxes

Same performance (**10 minutes**).

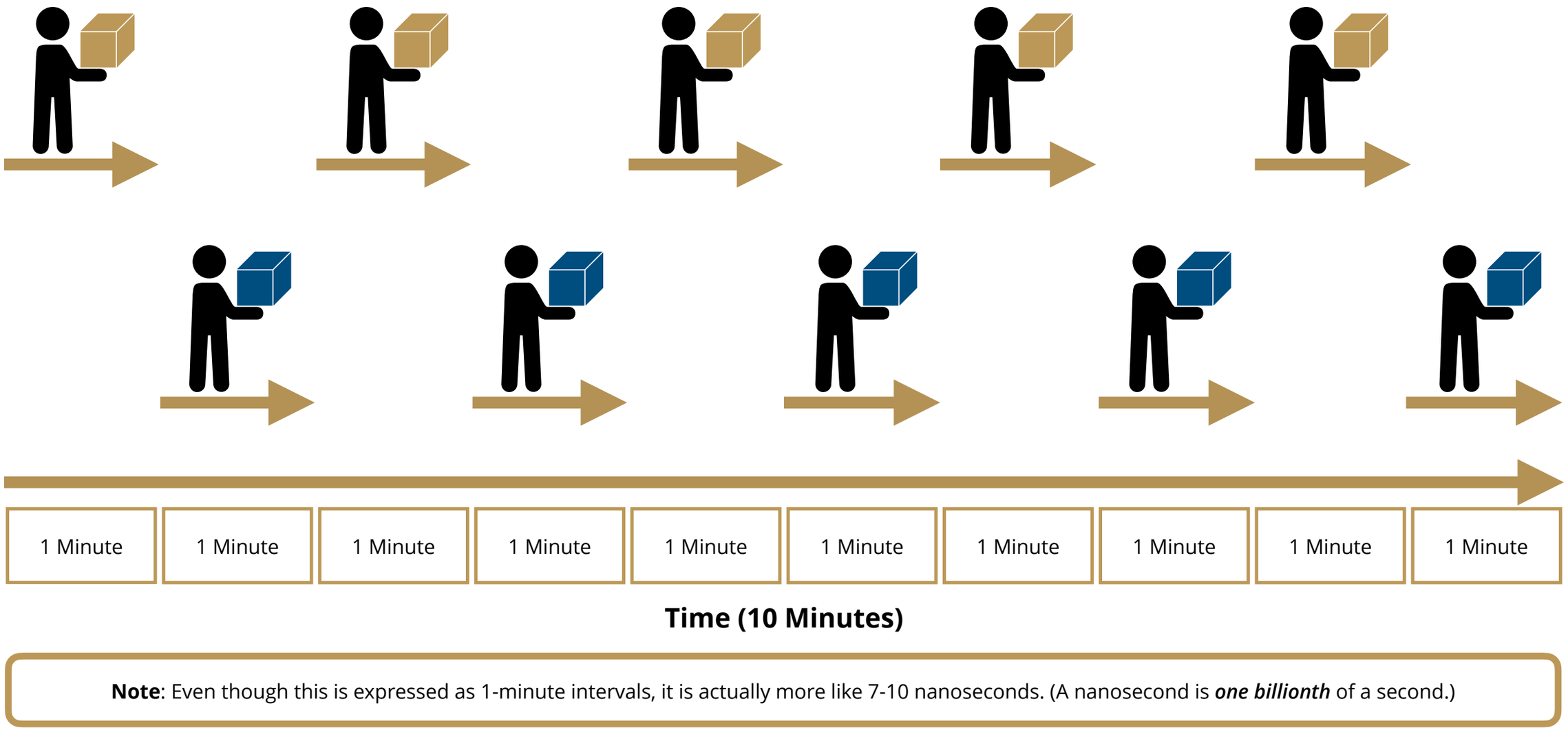
The advantage to doing it this way is you will start getting blue boxes sooner instead of waiting for ALL of the gold boxes to finish first.

2

Asynchronous Timing

2

Asynchronous work can also be expressed as tasks splitting up time, sharing time, or slicing up time to share between the tasks.
Here is another way to look at it:



Note: Even though this is expressed as 1-minute intervals, it is actually more like 7-10 nanoseconds. (A nanosecond is ***one billionth*** of a second.)

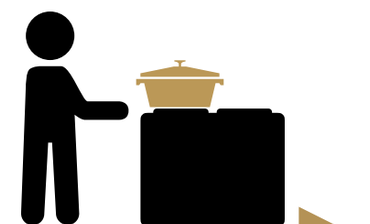
2

Asynchronous Timing Example

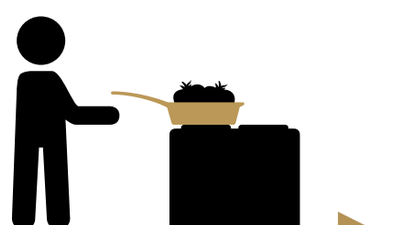
2

Here is another example of how performing asynchronous work might look like.
One person could start dividing their time to start cooking 3 different things at the same time.

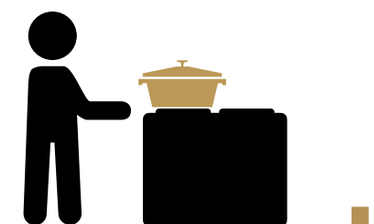
Task: Cook chicken



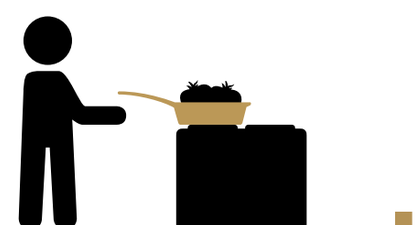
Task: Cook rice



Task: Cook vegetables



Task: Cook rice



Task: Cook vegetables

Task: Cook chicken

It will still take you 55 minutes. But the advantage is that the rice and vegetables will be completed before the chicken.

55 Minutes

2

Asynchronous Cooking

2

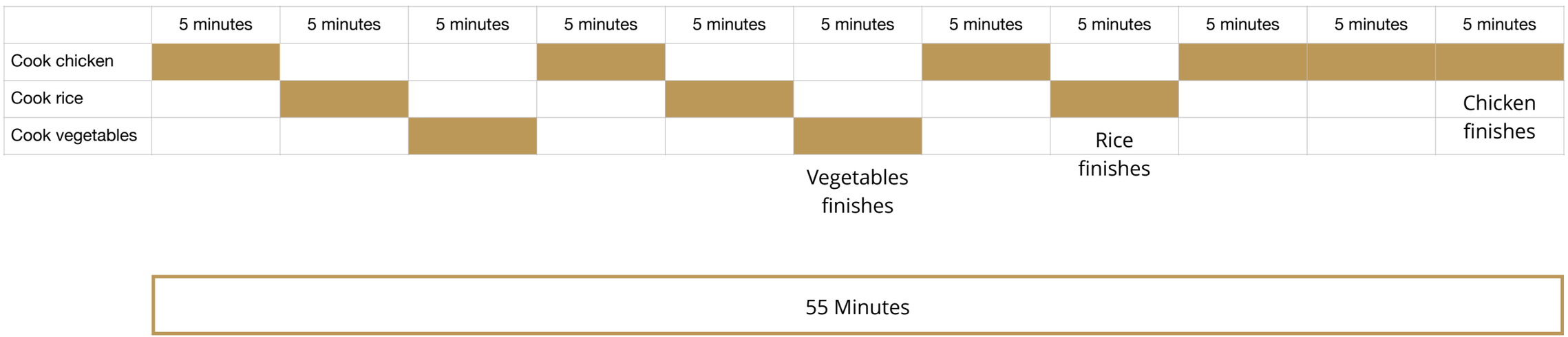
The previous page is not drawn to scale.

A more accurate example would really be **one stove to cook on (one core processor)**.

For example, the chef would put the chicken on for five minutes, take it off, put the rice on for five minutes, take it off and put the vegetables on for five minutes.

This would be repeated until one food is cooked and then it would keep rotating between the foods left.

This might be a more accurate image:



Parallel Work

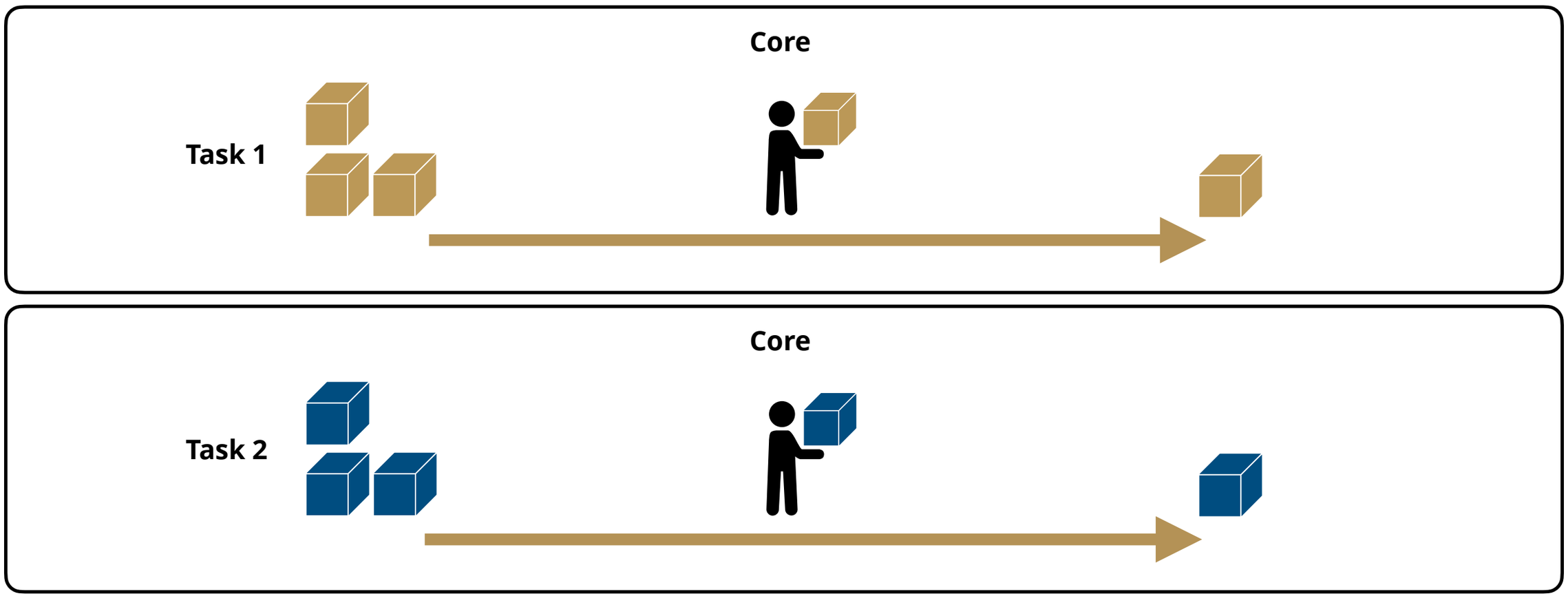
3

Parallel work means task 1 and task 2 can both start and execute at the same time.

Parallel Work

3

Parallelism (parallel work) is when two or more people can perform a task at the same time. Now there is a person assigned to move gold boxes and another for blue boxes. This is another form of concurrency (doing multiple things).



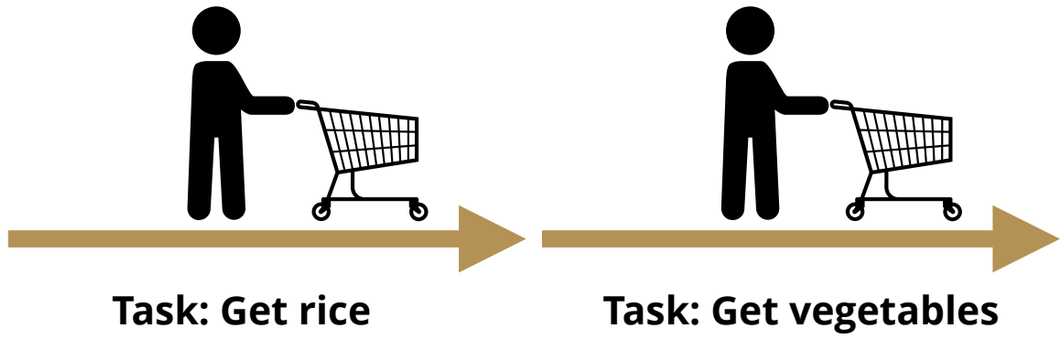
Note: The threads/people need to be on different cores to make parallelism work.

These people can move 10 boxes in **5 minutes**.
Better performance!

Parallel Work Example 1

3

If two people are shopping together, this might be a good time to use parallelism to get multiple things done at the same time.



Task: Get rice

Task: Get vegetables

Task: Get chicken

Task: Get cream

2 Minutes	2 Minutes

All shopping can now be done in **4 minutes** instead of 8 minutes.

3

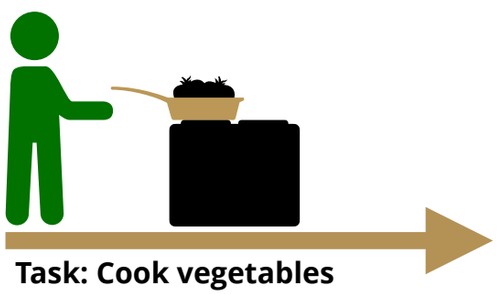
Parallel Work Example 2

3

In a busy restaurant, there might be 3 chefs all cooking at the same time. In this way, parallelism can also help increase efficiency.

Task: Cook chicken

Task: Cook rice



Task: Cook vegetables

30 Minutes

All 3 food will be cooked at the same time by 3 different people.

The lowest you can get the time down to is **30 minutes** because of how long it takes to cook the chicken.

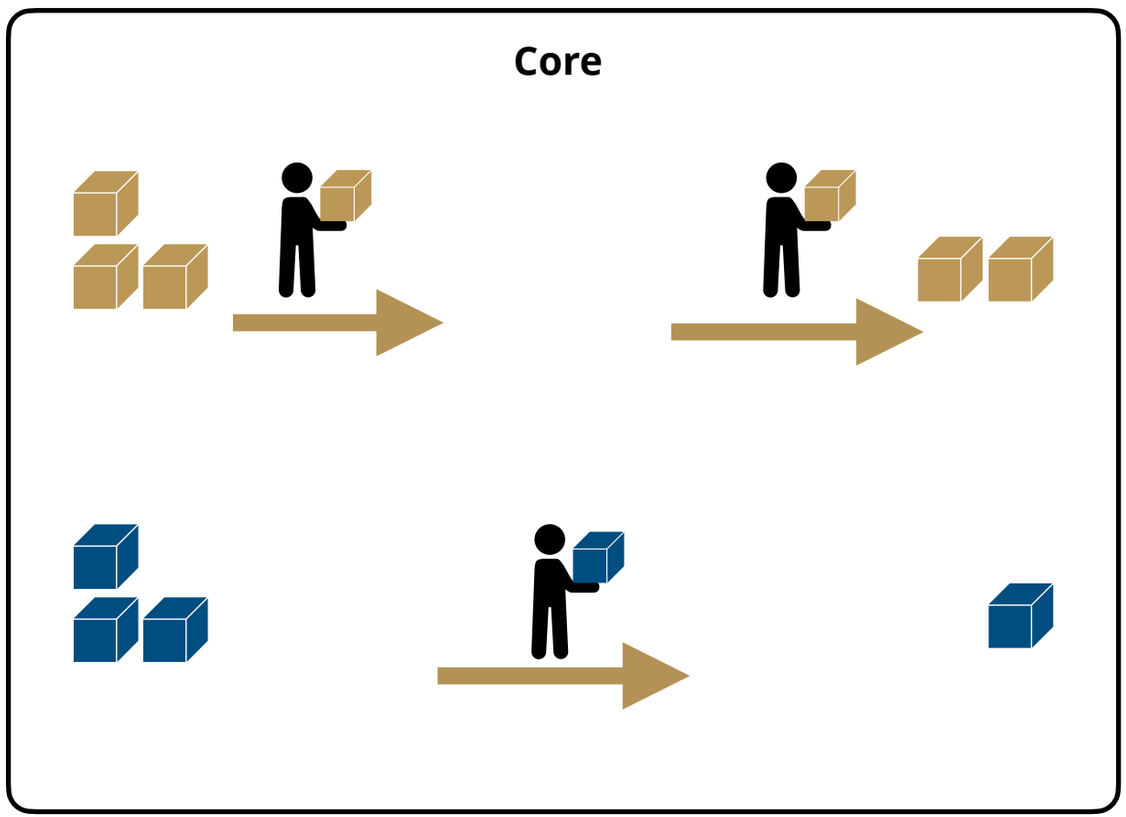
What is Concurrency?

So, where does the word “concurrency” come in?

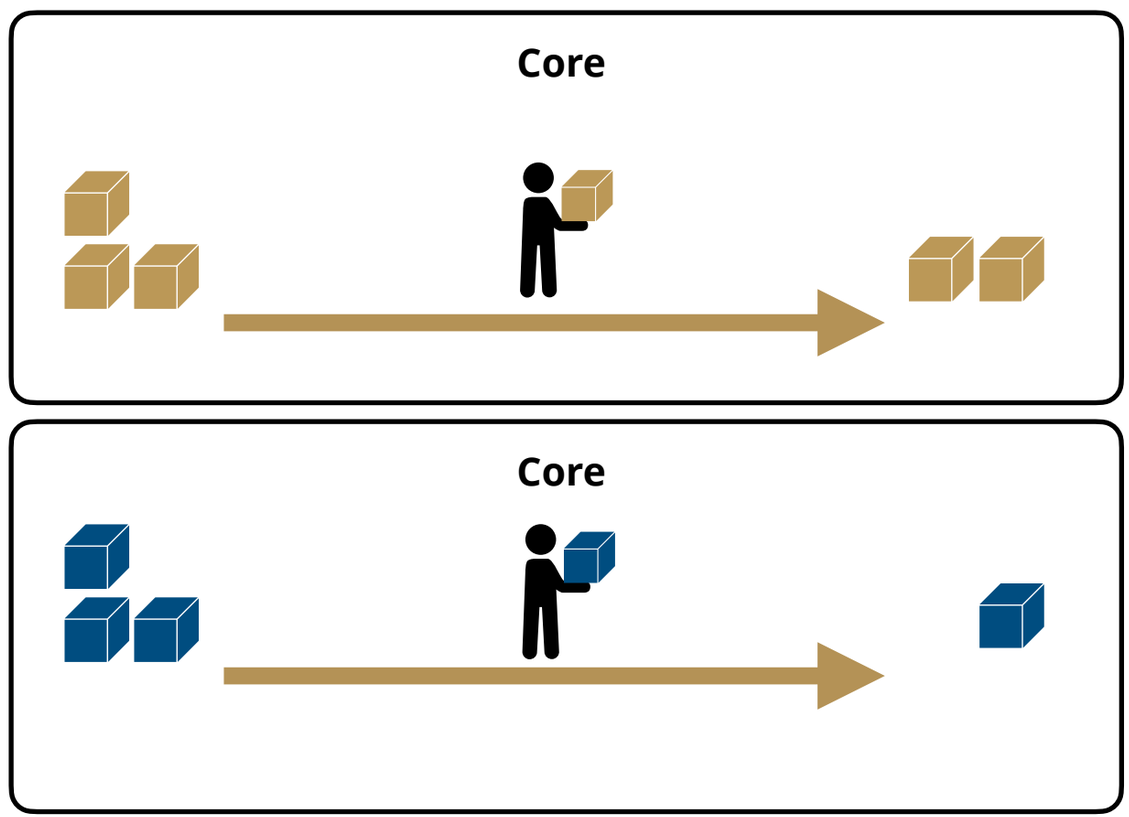
What is Concurrency?

The Swift language documentation on concurrency uses the term "**concurrency**" to refer to this common combination of asynchronous and parallel code. This book explores all the different ways you can create concurrent (asynchronous & parallel) code to perform tasks in your app.

Asynchronous Code



Parallel Code



The

End